8. 乱数とヒストグラム
(指定された分布に従う乱数)

2020/11/30

1 準備

講義で説明した、変換法及び棄却法によって指定された分布に従う乱数を生成しましょう。また、そのヒストグラムを作成し、指定した分布に従う乱数を生成できることを確認しましょう。先週配布したサンプルプログラムを利用します。

2 変換法

定義域が $[a, b)$ である確率密度関数 $f(x)$ に対する、確率分布関数 $F(x)$ は

$$F(x) = \int_a^x f(y)\mathrm{d}y \tag{2.1}$$

です。この確率分布関数 $F(x)$ の逆関数 $F^{-1}(x)$ が得られる場合、変換法を用いて確率密度関数 $f(x)$ に従う乱数を生成することができます。

クラス `method.Transform` は、前回用いた `Uniform` と同様に、`java.util.Random` の拡張クラスであり、変換法で指定された分布に従う乱数を生成します。確率分布関数の逆関数 $F^{-1}(x)$ は、`java.util.function.DoubleFunction` インターフェイスのインスタンスとして定義します。

`DoubleFunction<Double> invProDist` は、`double` 型の入力と戻り値 (`Double`) を指定した関数 `invProDist` の定義です。コンストラクタでは、$F^{-1}(x)$ を引数 `invProDist` として与えて、インスタンスを初期化します。

メソッド `netxDouble()` は、`java.util.Random.nextDouble()` によって一様乱数 $r \in [0, 1)$ を生成し、$F^{-1}(r)$ を戻り値として返します。`DoubleFunction<Double> invProDist` は、インターフェイスのインスタンスであり、関数そのものではないことに注意します。実際に使用する具体的な関数は、そのメソッド `apply()` です。

課題 1　メソッド `netxDouble()` 内の記述を確認しましょう。

具体例として、指数分布を考えます。`samples.Exp` クラスの `main()` において、確率分布関数の逆関数 $F^{-1}(x)$ を定義しています。配布した例では、定義域 $[0, 1)$ の指数確率密度

$$f(x) = Ae^{-x}, \quad A = \frac{e}{e-1} \tag{2.2}$$

に対応した、確率分布関数の逆関数

$$F^{-1}(r) = -\ln(1 - r/A) \tag{2.3}$$

を用いています。なお、Java 中の `Math.log(x)` は自然対数 $\ln(x)$ を、`Math.E` は自然対数の底 e を数値的に与えます。

課題 2　式 (2.2) で定義する指数関数分布に従う乱数を生成しなさい。その結果を gnuplot を用いて、期待される確率密度関数 (式 (2.2)) とともに図示することで、正しく乱数が生成できることを示しなさい。gnuplot 中で定数 e(自然対数の底) を数値的に与えるためには `exp(1)` を用います。

課題 3　次は、確率密度関数が

$$f(x) = \frac{1}{2}\sin(x), \quad x \in [0, \pi) \tag{2.4}$$

の場合を考えます。対応する確率分布は

$$F(x) = \frac{1}{2}\left(1 - \cos(x)\right) \tag{2.5}$$

であり、その逆関数は

$$F^{-1}(r) = \cos^{-1}\left(1 - 2r\right) \tag{2.6}$$

です。ここで $\cos^{-1}(x)$ は $\cos(x)$ の逆関数 arc cosine であり、Java では `Math.acos()` に対応します。クラス `Exp` を参考に、`main()` のみのクラス `Sin` を作成し、確率分布 (式 (2.4)) に従う乱数を生成しなさい。その結果を gnuplot で式 (2.4) とともに表示し、正しく乱数を生成することを示しなさい。

3　棄却法

確率密度関数 $0 \leq f(x) < m$ の不定積分である確率分布関数 $F(x)$ が得られ、さらにその逆関数が得られることは非常に稀な場合です。変換法が使えない場合に、指定した分布に従う乱数を生成するために用いるのが棄却法です。

棄却法では、確率密度関数の定義域 $[a, b)$ に一様乱数 x を生成し、もう一つの一様乱数 $y \in [0, 1)$ が $y < f(x)/m$ ならば x を採用し、それ以外の場合には棄却することで、確率分布関数 $f(x)$ に従う乱数を生成する方法です。棄却した場合には、次の一様乱数の組 (x, y) を生成して、乱数を得るまで上記を繰り返します。

課題 4　`method.Rejection` は、`java.util.Random` の拡張クラスであり、棄却法で乱数を生成します。コンストラクタには、乱数の範囲、確率密度関数、及びその最大値を与えます。メソッド `nextDouble()` を完成させなさい。

配布した `samples.SinSquare` クラスの `main()` メソッドでは、確率密度関数が

$$f(x) = \frac{2}{\pi}\sin^2(x), \quad x \in [0, \pi) \tag{3.1}$$

である場合を記述しています。Java では、`Math.sin()` は正弦関数を、`Math.PI` は π を表します。

課題 5　式 (3.1) で定義する三角関数の二乗の確率密度に従う乱数を生成なさい。その結果を gnuplot を用いて、期待される確率密度関数とともに図示することで、正しく乱数が生成できることを示しなさい。また、gnuplot では `pi` が π を数値的に与えます。

課題 6　第二の例として、定義域 $x \in [0, 1)$ で定義する以下の確率密度関数を考えましょう。

$$f(x) = \begin{cases} 2/3 & \text{if } x < 1/2 \\ 4/3 & \text{otherwise} \end{cases} \tag{3.2}$$

式 (3.2) で定義する確率密度に従う乱数を生成しなさい。そのために、`samples.SinSquare` クラスを参考に、`samples.Step` クラスを作成しなさい。実行の結果を gnuplot を用いて図示することで、正しく乱数が生成できることを示しなさい。